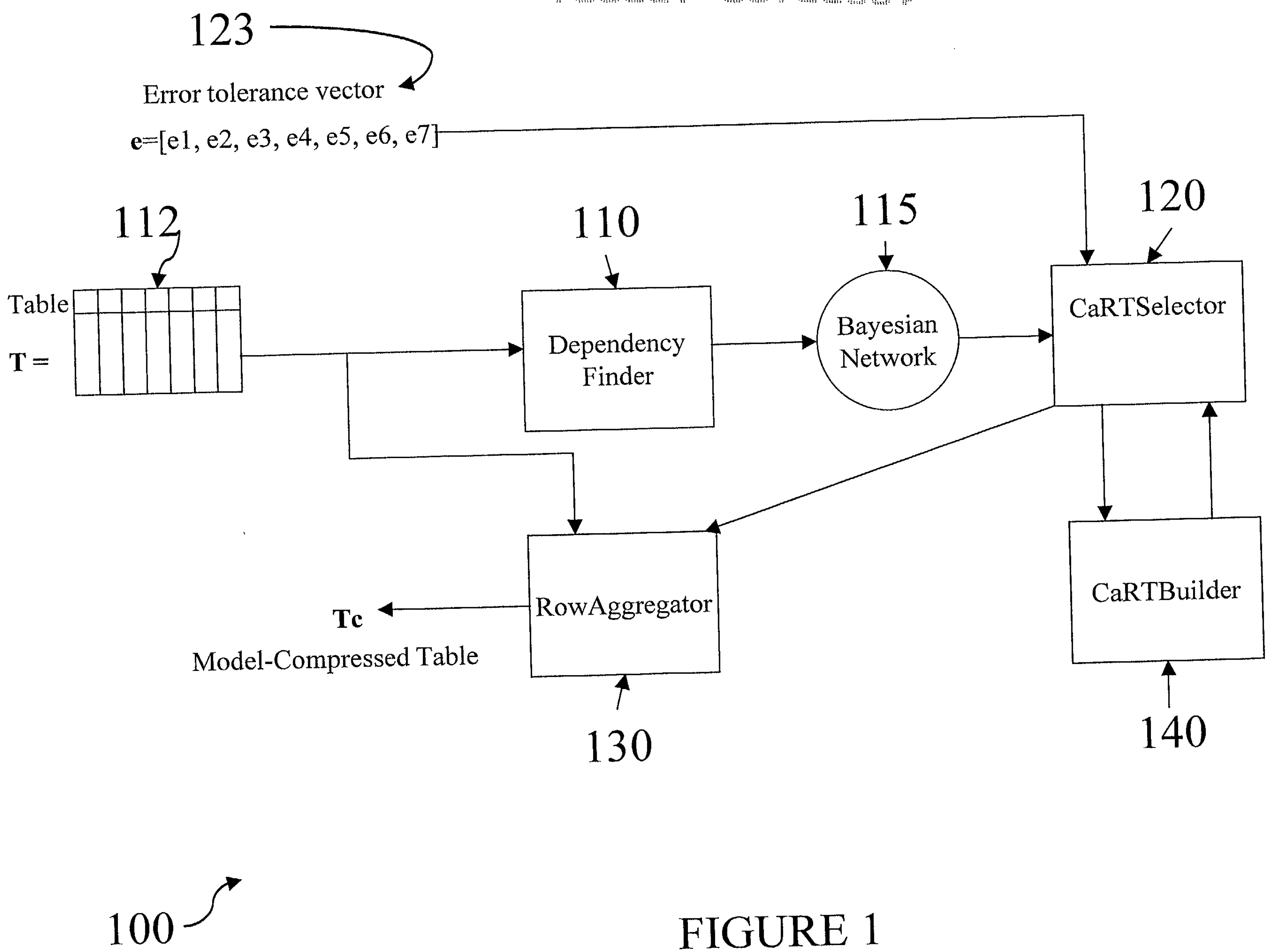

123
Error tolerance vector
e=[e1, e2, e3, e4, e5, e6, e7]
112
Table
T =
110
Dependency Finder
115
Bayesian Network
120
CaRTSelector
RowAggregator
Tc
Model-Compressed Table
130
CaRTBuilder
140
100

FIGURE 1

procedure Greedy $(T(X), \bar{e}, G, \theta)$

Input: n-attribute table T and n-vector of error tolerances $\bar{e}$; Bayesian network G on the set of attributes X and threshold θ on the relative benefit for selecting a CaRT predictor.

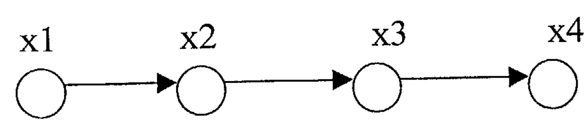
Output: A set of materialized (predicted) attributes X_{mat} ($X_{pred} = X - X_{mat}$) and a CaRT predictor for each $X_i \in X_{pred}$.

begin

1. $X_{mat} := X_{pred} := \phi$
2. let $< X_1, X_2, \ldots, X_n >$ be the attributes in X sorted in *topological order* of G
3. **for i** $:= 1, \ldots, n$
4. if $\pi(X_i) = \phi$ then $X_{mat} := X_{mat} \cup \{X_i\}$ /* X_i must be materialized if it has no parents in G */
5. else
6. $M :=$ BuildCaRT $(X_{mat} \rightarrow X_i,\ e_i)$
7. if (MaterCost (X_i) / PredCost $(X_{mat} \rightarrow X_i) > \theta$) then $X_{pred} := X_{pred} \cup \{X_i\}$
8. else $X_{mat} := X_{mat} \cup \{X_i\}$
9. end
10. end

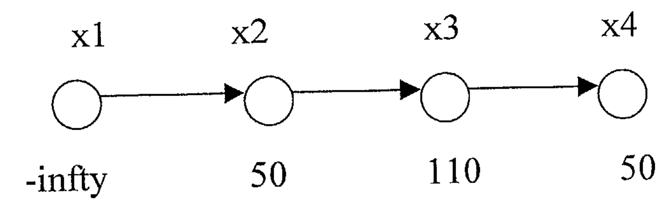
end

FIGURE 2: The Greedy CaRT Selection Algorithm



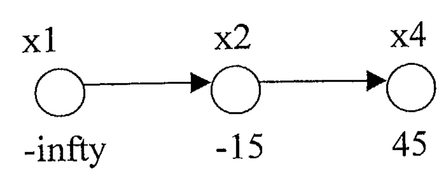
Bayesian Network G

FIGURE 3A



(b) Gtemp (first iteration)

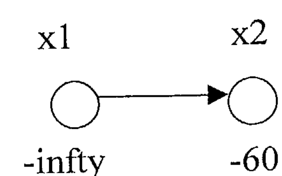
FIGURE 3B



Gtemp (second iteration)

FIGURE 3C

FIGUREs 3A-3D



Gtemp (third iteration)

FIGURE 3D

```
procedure MaxIndependentSet (T(X), ē, G, neighborhood() )
Input:    n-attribute table T and n-vector of error tolerances ē;
          Bayesian network G on the set of attributes X and function
          neighborhood()defining the "predictive neighborhood" of a
          node X_1 in G (e.g., π (X_1) or β (X_1)).
Output:   A set of materialized (predicted) attributes X_mat (X_pred = X -
          X_mat) and a CaRT predictor PRED (X_i) → X_i for each X_i  ∈ X_pred.
begin
    X_mat := X, X_pred := ϕ
    PRED(X_i) := ϕ for all X_i ∈ X, improve := true
    while (improve ≠ false) do
        for each X_i ∈ X_mat
            mater_neighbors (X_i) :=
            (X_mat ∩ neighborhood(X_i))∪{PRED(X):X ∈ neighborhood
            (X_i), X ∈ X_pred}-{X_i}
            M := BuildCaRT (Mater_neighbors (X_i)→X_i, e_i)
            let PRED (X_i) ⊆ mater_neighbors (X_i) be the set of
            predictor attributes used in M
            cost_change_i :=0
            for each X_j ∈ X_pred such that X_i ∈ PRED(X_j)
                NEW_PRED_i (X_j) := PRED(X_j)-{X_i}∪PRED(X_i)
                M :=BuildCaRT (NEW_PRED_i(X_j)→X_j, e_j
                set NEW_PRED_i (X_j) to the (sub)set of
                predictor attributes used in M
                cost_change_i := cost_change_i + (PredCost(PRED
                (X_j)→X_j)-PredCost (NEW_PRED_i(X_j)→X_j))
            end
        end
        build an undirected, node-weighted graph G_temp = (X_mat,
        E_temp) on the current set of materialized
        attributes X_mat, where:
            (a) E_temp := {(X,Y) : for all pairs X, Y ∈ X_pred}∪
                                    {(X_i,Y): for all Y ∈ X_mat}
            (b) weight (X_1) := MaterCost (X_i) -PredCost (PRED(X_i)
            → X_i)+cost_change_i for each X_i ∈ X_mat
        S := FindWMIS (G_temp)    /* select (approximate) maximum
        weight independent set in G_temp
                                    as "maximum-benefit" subset of
                                    predicted attributes  */
        if (∑_{X∈S} weight (X) ≤ 0) then improve := false
        else/* update X_mat, X_pred, and the chosen CaRT predictors */
            for each X_j ∈ X_pred
                if (PRED(X_j)∩ S = {X_i}) then PRED (X_j) :=
                NEW_PRED_i(X_j)
            end
            X_mat := X_mat - S, X_pred := X_pred ∪ S
        end
    end /* while */
end
```

FIGURE 4: The MaxIndependentSet CaRT Selection Algorithm

```
procedure LowerBound (N, e_i, b)
Input:    Leaf N for which lower bound on subtree cost is to be
          computed; error tolerance e_i for attribute X_i; bound b
          on the maximum number of internal nodes in subtree
          rooted at N.
Output:   Lower bound L(N) on cost of subtree rooted at N.
begin
     for i := to r
          minOut [i,0] :=i
     for J := 1 to b + 1
          minOut[0,j] :=0
     l :=0
     for i := 1 to r
          while x_i - x_{l+1} > 2_ei
          l :=l = 1
      for j := 1 to b + 1
          minOut[i,j] := min {minOut[i - 1,j] + 1, minOut [l,j-1]
     end
     L(N) := ∞
     for J := 0 to b
          L(N) :=min {L(N), 2j + 1 + j log (|X_i|)+ (j + 1 + minOut
           (r,j+1)) log (|dom(X_i)|)}
     L(N) := min {L(N), 2b + 3 + (b + 1) log (|X_i|)+ (b + 2) log
     (|dom(X_i)|)}
     return L(N)
end
```

FIGURE 5: Algorithm for Estimating Lower Bound on Subtree Cost

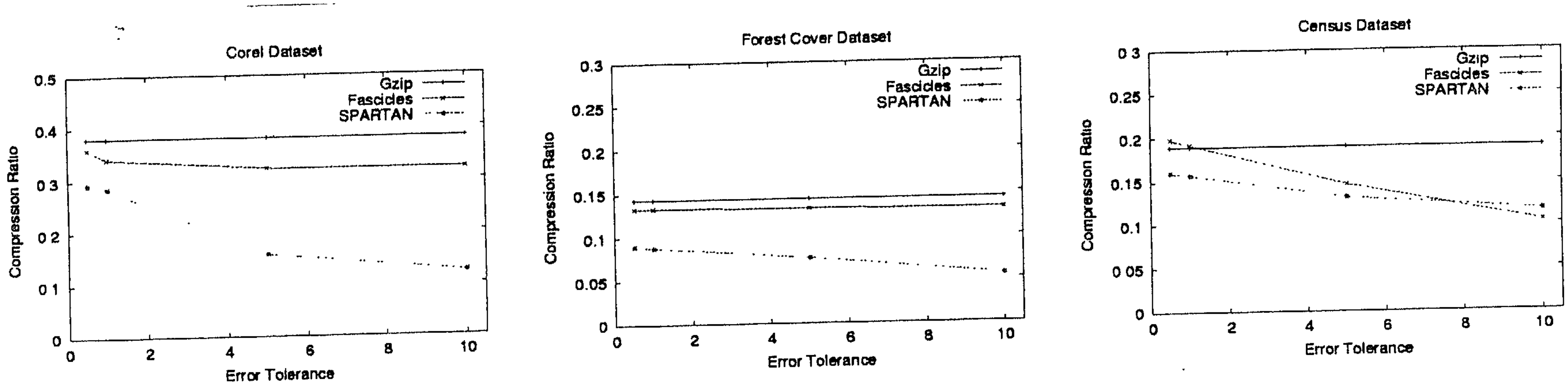
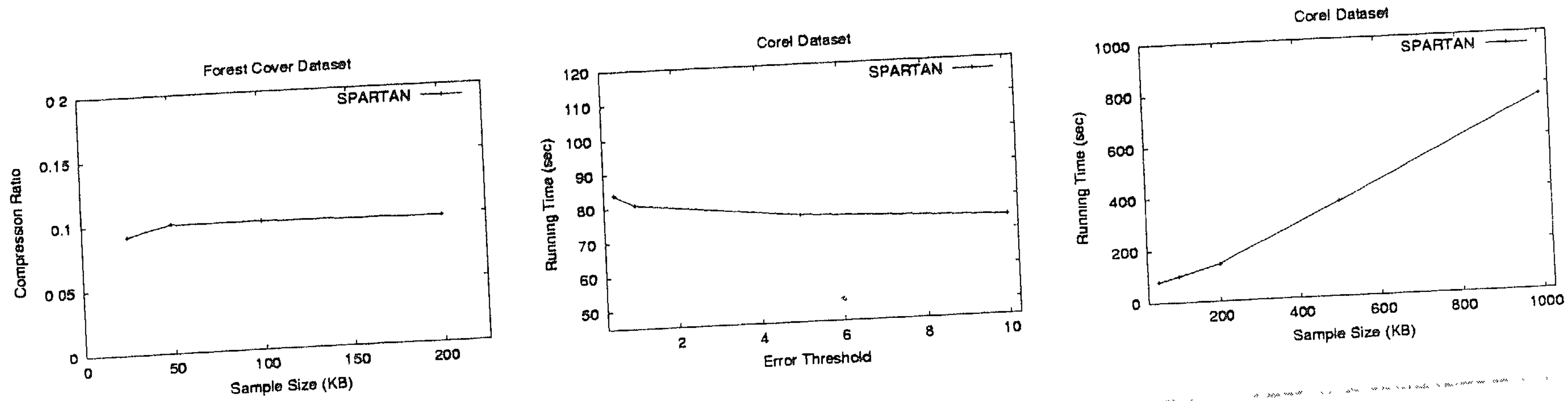

Corel Dataset
Forest Cover Dataset
Census Dataset
Gzip
Fascicles
SPARTAN
Compression Ratio
Error Tolerance
0
0.1
0.2
0.3
0.4
0.5
0.05
0.15
0.25
2
4
6
8
10

FIGURE 6



Forest Cover Dataset
SPARTAN
Compression Ratio
0 2
0.15
0.1
0 05
0
0
50
100
150
200
Sample Size (KB)
Corel Dataset
SPARTAN
Running Time (sec)
120
110
100
90
80
70
60
50
2
4
6
8
10
Error Threshold
Corel Dataset
SPARTAN
Running Time (sec)
1000
800
600
400
200
0
200
400
600
800
1000
Sample Size (KB)

FIGURE 7.